Domain-Specific Languages for implementing Variability

voelter@acm.org
www.voelter.de
@markusvoelter
Markus Völter

itemis Voelter { ingenieurbüro für softwaretechnologie //

1

The Big Picture

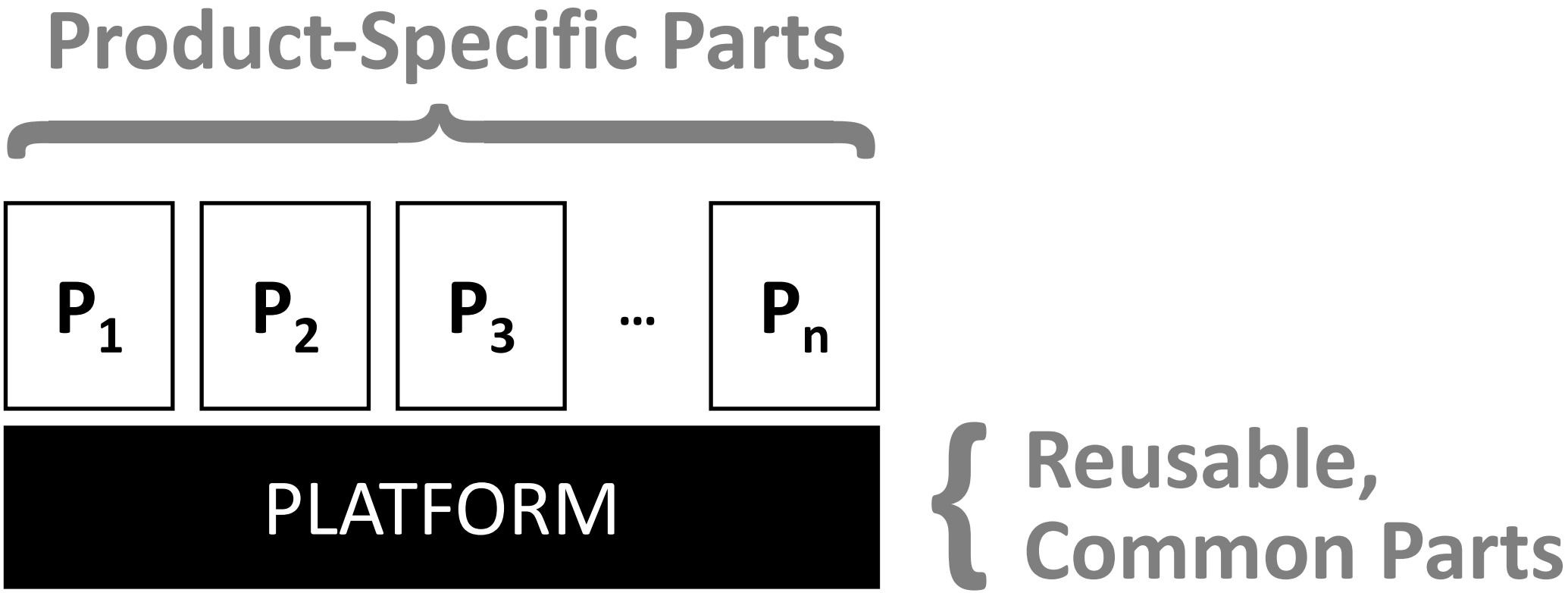
THE ESSENCE OF PRODUCT LINES

Efficiently implement a set of **related** products (for different markets or other contexts), and **evolve** all of them efficiently over time.

Reuse **common** parts.

Be productive in the **differences**.

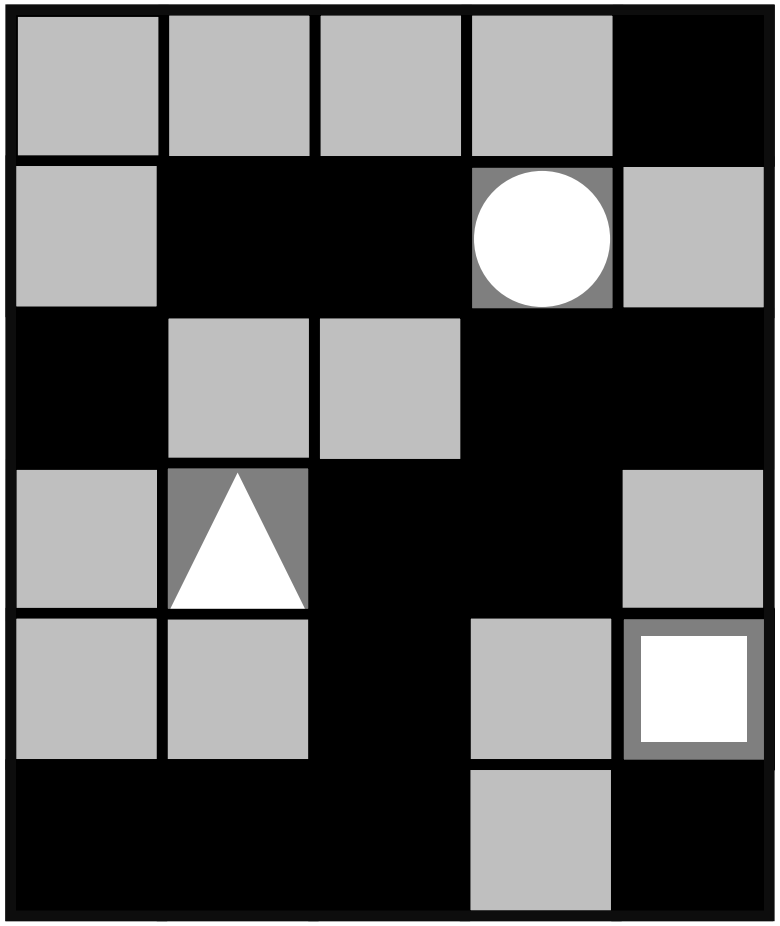
THE ESSENCE OF PRODUCT LINES

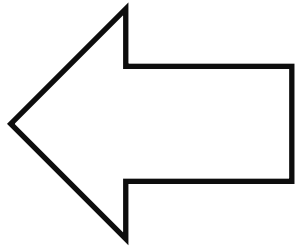


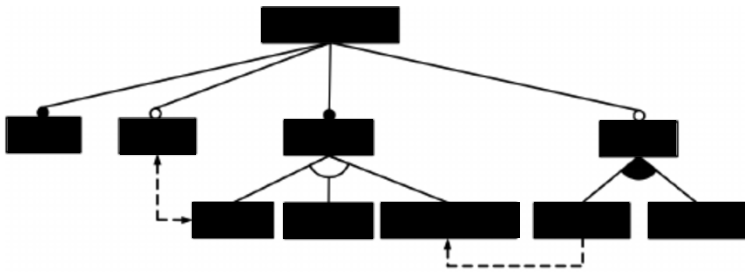
2

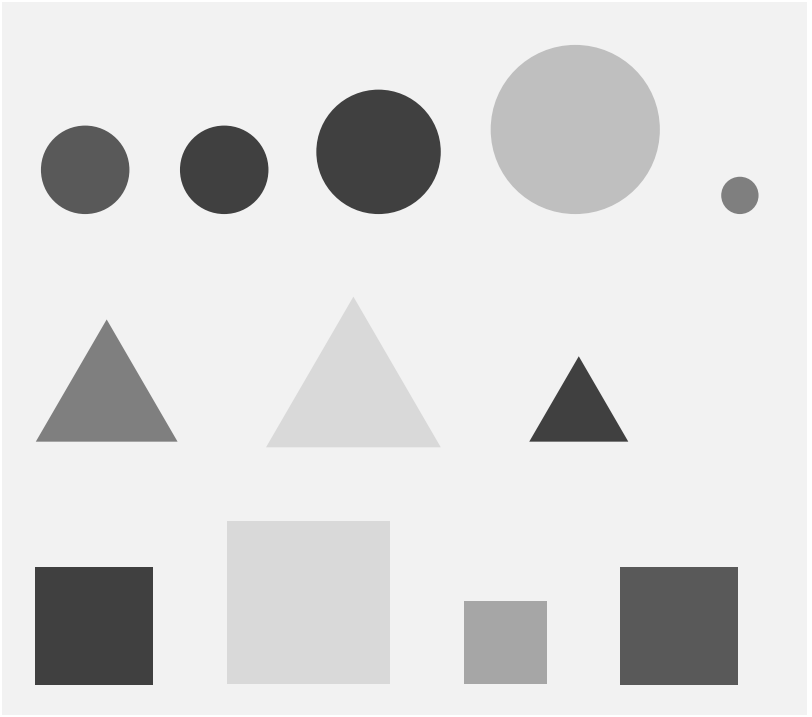
Configuration vs. Construction

CONFIGURATION



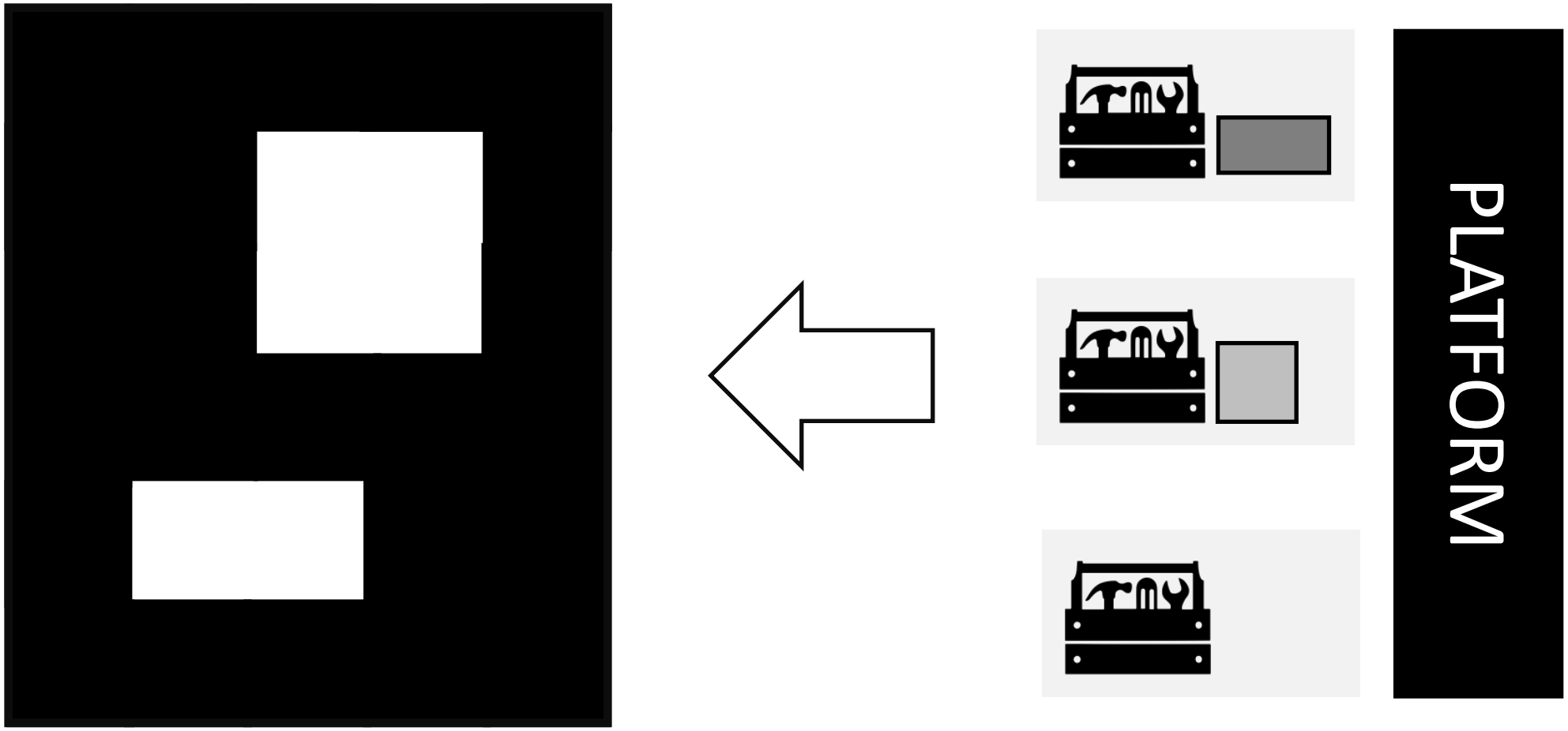




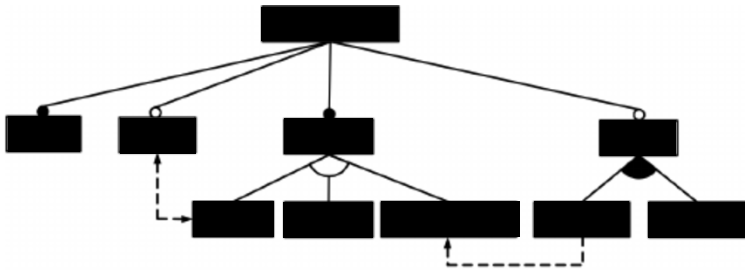


PLATFORM

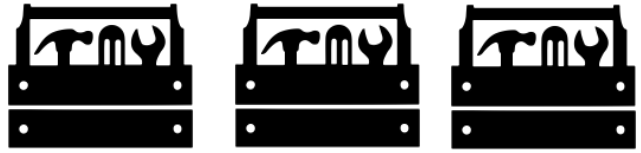
CONSTRUCTION



COMPARISON



CONFIGURATIVE



CONSTRUCTIVE

Platform = Artifacts
"Selection"
Limited Set of Options
Easy to "assemble" product.

Platform = Tools
"Building"
Unlimited Set of Options.
More effort per Product.

3

Domain-Specific Languages

“A domain-specific language (DSL) is a computer language **specialized** to a particular application domain.”

Abstractions/Metamodel
Notation/Syntax
Analyses + Error Reporting
IDE + other Tools

berbwvekFF (lkm_akt_param; lkm_faell_param; ber_zweck_param; kz_rzw_param) ×

Funktionenmodell berbwvekFF

Formale Beschreibung

Funktion: berbwvekFF
Programmquelle: vmsctfa1.c
Produkt-Typ: Fonds **PK-Typ:** Kapital-Konto
Status: 18.1

Parameter-Attribute

lkm_akt_param
lkm_faell_param
ber_zweck_param
kz_rzw_param

Verwendete VADM-Attribute

Keine verwendeten VADM-Attribute, werden automatisch hinzugefügt

Rückgabe-Attribut

bwvek

aufgerufene Funktionen

VTRKermbtgfaellFF (a)
berbweinzelFF (a; b; c)

Beschreibung

Die Funktion liefert den Barwert per @lkm_akt_param des vorschüssigen Zahlungss
@lkm_akt_param bis @lkm_faell_param – jeweils einschließlich. Zahlungszeitpunkt
#lkm_akt_param – 1# bis #lkm_faell_param – 1#. Der Parameter @kz_rzw_param steu
Zahlweise des Zahlungsstroms. Möglich sind zur Zeit nur die Ausprägungen 0 (Zahl
und 12 (monatliche Zahlungsweise).

Hilfsvariablen

kz_bf_hilf

Verarbeitungen

```
Schleife über lkm_faell_hilf = lkm_akt_param bis lkm_faell_param
  Falls kz_rzw_param = 12
      kz_bf_hilf = 1
    sonst
      kz_bf_hilf = VTRKermbtgfaellFF (lkm_faell_hilf)
  Ende Falls kz_rzw_param = 12
  bwvek = bwvek + kz_bf_hilf * berbweinzelFF (lkm_akt_param; lkm_faell_hilf - 1
Ende Schleife über lkm_akt_param bis lkm_faell_param

return bwvek
```

Formale Beschreibung

Funktion:	berbwvekFF		
Programmquelle:	vmsctfa1.c		
Produkt-Typ:	FONDS	**PK-Typ:**	KAPITAL-KONTO
	Name	Verw.	Entität
verwendete Attribute:	lkm_akt_param	E	PARAMETER
	lkm_faell_param	E	PARAMETER
	ber_zweck_param	E	PARAMETER
	kz_rzw_param	E	PARAMETER
	bwvek	A	RETURN
aufgerufene Funktionen:	berbweinzelFF		
	VTRKermbtgfaellFF		
Status:	18.1		

Verarbeitungen

Die Funktion liefert den Barwert per lkm_akt_param des vorschüssigen Zahlungsstroms der Höhe 1 von Monat lkm_akt_param bis lkm_faell_param – jeweils einschließlich. Zahlungszeitpunkte sind jeweils die Monatsbeginne, also **lkm_akt_param** -1 bis **lkm_faell_param** – 1.

Der Parameter **kz_rzw_param** steuert die zu berücksichtigende Zahlweise des Zahlungsstroms. Möglich sind zur Zeit nur die Ausprägungen 0 (Zahlungen zu den Beitragsfälligkeiten) und 12 (monatliche Zahlungsweise).

Schleife über lkm_faell_hilf = **lkm_akt_param** bis **lkm_faell_param**

Falls **kz_rzw_param** = 12

kz_bf_hilf = 1

sonst

kz_bf_hilf = *VTRKermbtgfaellFF*(lkm_faell_hilf)

Ende Falls **kz_rzw_param** = 12

bwvek = **bwvek**

+ kz_bf_hilf * *berbweinzelFF*(**lkm_akt_param**, lkm_faell_hilf - 1, **ber_zweck_param**)

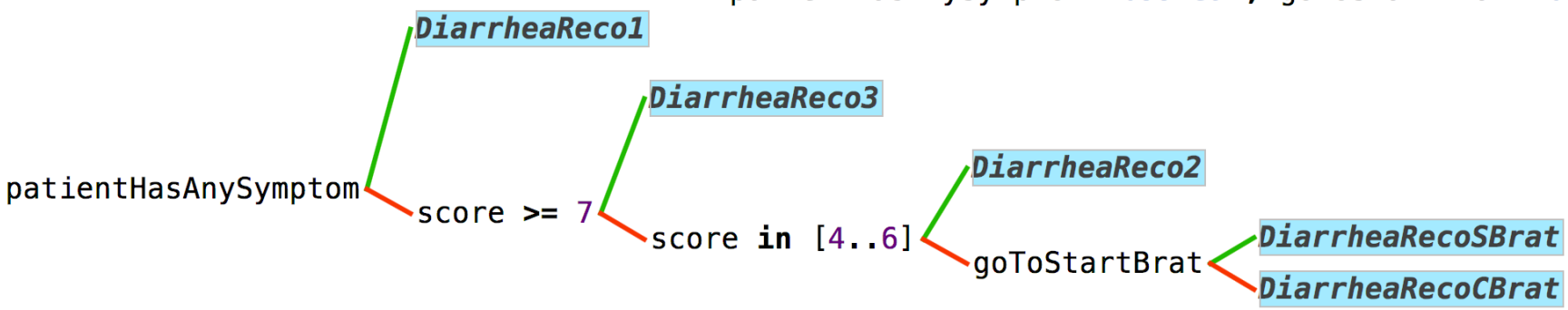
Ende Schleife

return **bwvek**

```
decision table BpScoreDecisionTable(sys: bpRange, dia: bpRange) =
```

		dia					
		<= 50	[51..90]	[91..95]	[96..100]	[101..109]	>= 110
sys	<= 90	1	1	3	4	5	6
	[91..140]	2	2	3	4	5	6
	[141..150]	3	3	3	4	5	6
	[151..160]	4	4	4	4	5	6
	[161..179]	5	5	5	5	5	6
	>= 180	6	6	6	6	6	6

```
decision tree DiarrheaStoolsDecisionTree(score: DiarrheaStoolsOverBaseline,
                                         patientHasAnySymptom: boolean, goToStartBrat: boolean)
```




```
type temperature: number[36|42]{1}
type measuredTemp: number[35|43]{2}

val T_measured: measuredTemp = 42.22
val T_calibrated: temperature = T_measured * 0.93
```

Error: type number[32.55|39.99]{4} is not a subtype of number[36|42]{1}

D : Kommutationswerte

Ergebnistyp:	Laufvariable:	Parameter:
number{3}	x	i geschlecht q

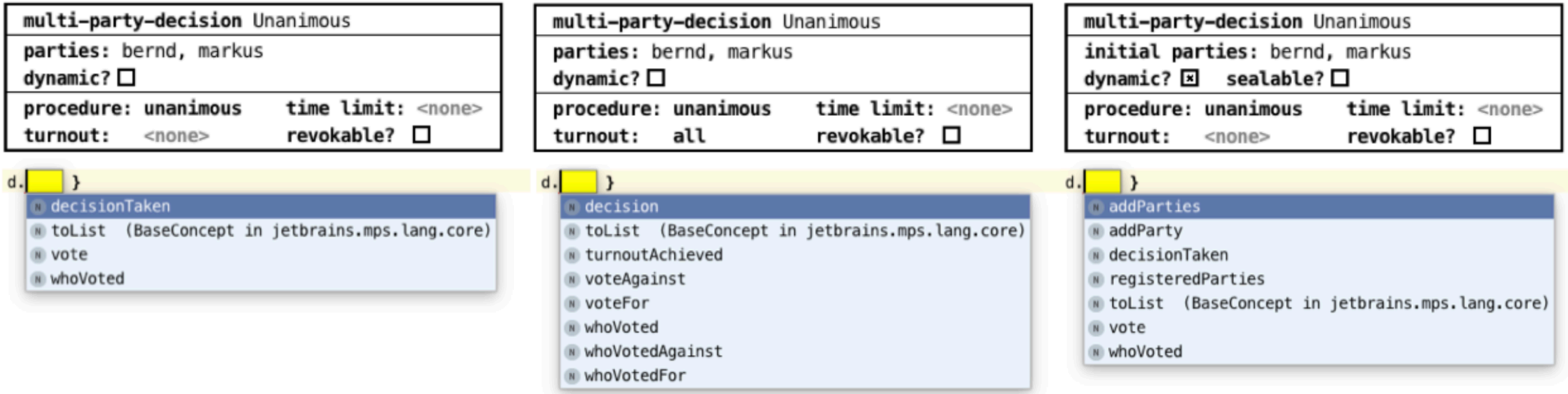
$$D_x := l_x * \frac{1}{(1 + i)^x}$$

l : Lebende im Jahr x

Ergebnistyp:	Laufvariable:	Parameter:
number{0}	x	geschlecht q

l_0 := startwertLebende

$l_x := l_{x-1} * (1 - q.\mathbf{lookup}(x, \text{geschlecht}))$




```
state machine Game (initialPlayers: list<party>) {
  var players = initialPlayers
  var pendings = nopendings
  var bids = nobids
  var lastWinner: opt<party> = none

  event requestAccess
  event startVoting
  event voteCandidate(candidate: party, vote: boolean)

  event terminatePendings
  event play
  event offerBid(money: bid)
  event endGame

  fun clearGame/RM() {
    lastWinner := none
    bids := nobids
  }

  state playing [senderIs(players)] {
    entry: clearGame/M(EMPTY )
    state bidding [takeTurns(players|ordered|after 500 boot out)] (initial) {
      on offerBid(money) : bids := bids.put(sender->money)
      if [timeInState > 2000] -> finished
    }
    state finished {
      entry: {
        val maxOfferedValue = bids.values.max
        lastWinner := bids.keys.findFirst(bids[it] == maxOfferedValue)
      }
      on endGame -> setup
    }
    exit: clearGame/M(EMPTY )
  }
}
```

Name: UVG-Leistungen

Unterhaltsvorschuss

Zeitangabe: laufend
Häufigkeit: monatlich einmal
Leistungskontext:
Leistungsart: Leer
Zählart: uvg
Anspruch Beginn: Anfang – Unbegrenzt: junger Mensch.geburtsdatum

Anspruch Ende: 01.01.1800 – 31.12.9999 **: min(junger Mensch**.geburtsda
12 Jahre

Zeitraum für Berechnung: Anfang – Unbegrenzt: {standardzeitraum, standardz

zweckgebundene Leistung: ☐
dem Grunde nach: ☐

Zeitraumbezogene Daten

nullwerte Anzeigen **: boolean** = 01.01.1800 – 31.05.2016 **: true**
01.06.2016 **– Unbegrenzt: false**

berechnungsart **: berechnungsarttyp** = 01.01.1800 – 31.12.9999 **:**

Bezugsobjekte: << ... >>
Attribute:
bemerkung **: string** wird validiert
antragsdatum **: Datum**

Nebenberechnungen

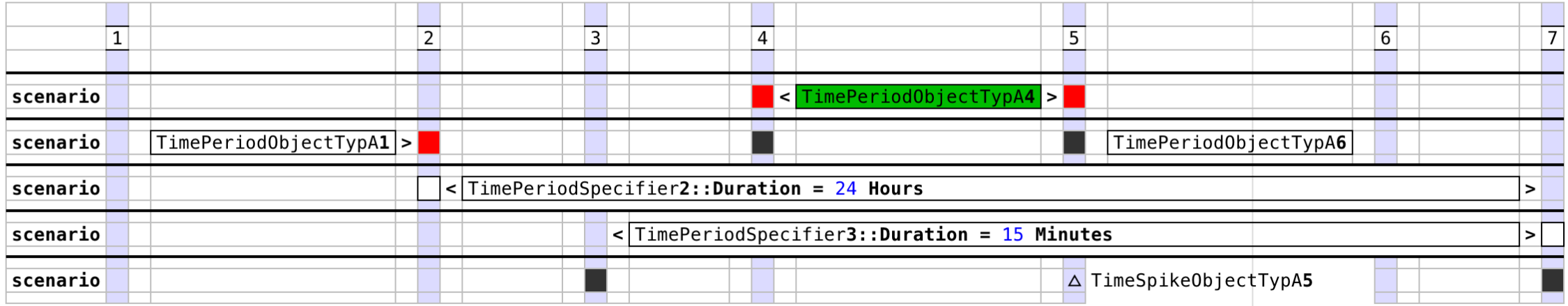
Name: Kindergeld für vollen Monat

(01.01.1800 – 31.12.9999 **)**

Rechnungsart: wenn: wird geboren **mit junger Mensch als** person **dann voller**
sonst: taggenau
Begünstigtenprinzip: ☐
monatswert = Kindergeld 1. Kind

zwischenergebnisse = [<< ... >>]
endergebnis = monatswert

```
rule TimingPattern_07_ObjectsAcrossRows_Modified_TandA_897_1Iteration {
description: 1381R3.ARCHD.0609.TimingPatternLanguage_TestScenarioMap.xlsx

parameters: TimePeriodObjectTypA4

patterns:
```




```
}
```

database databaseOneElementAcrossRows

Type	Begin	End	Duration	Occurence
eTimePeriodObjectTypA	500	550	50	

database databaseOneAndMoreIterationsHappy

Type	Begin	End	Duration	Occurence
eTimePeriodObjectTypA	50	100	50	
eTimeSpikeObjectTypA				86000
eTimePeriodObjectTypA	86020	86030	10	

Risks Table		Required attack potentials				
		Beyond high	High	Moderate	Low	Very low
Damage potentials	Very low	Low risk	Low risk	Low risk	Moderate risk	Moderate risk
	Low	Low risk	Low risk	Moderate risk	High risk	High risk
	Moderate	Low risk	Moderate risk	High risk	High risk	Very high risk
	High	Moderate risk	High ris			
	Very high	Moderate risk	High ris			

SZ.42
Integrity Webservice
RAP: Very low

SPEX	ACCS	TIME	EQUIP	KNOW	eRAP
PRFC	RMAC	HRS	STRD	PUBL	
PRFC	RMAC	HRS	STRD	PUBL	

Manipulation/Tampering...
RAP: Very low

SPEX	ACCS	TIME	EQUIP	KNOW	eRAP
PRFC	RMAC	HRS	STRD	PUBL	
PRFC	RMAC	HRS	STRD	PUBL	
Independent: Very low					
PRFC	RMAC	HRS	STRD	PUBL	
PRFC	RMAC	HRS	STRD	PUBL	

\+

B.14
(Identity) Spoofing of...
RAP: Beyond high

SPEX	ACCS	TIME	EQUIP	KNOW	eRAP
PRFC	RMAC	HRS	STRD	RSTR	Beyond high
PRFC	RMAC	HRS	STRD	RSTR	Beyond high
Independent: Very low					
PRFC	RMAC	HRS	STRD	RSTR	
PRFC	RMAC	HRS	STRD	RSTR	

\+

B.15
Man-in-the-Middle Atta...
RAP: Beyond high

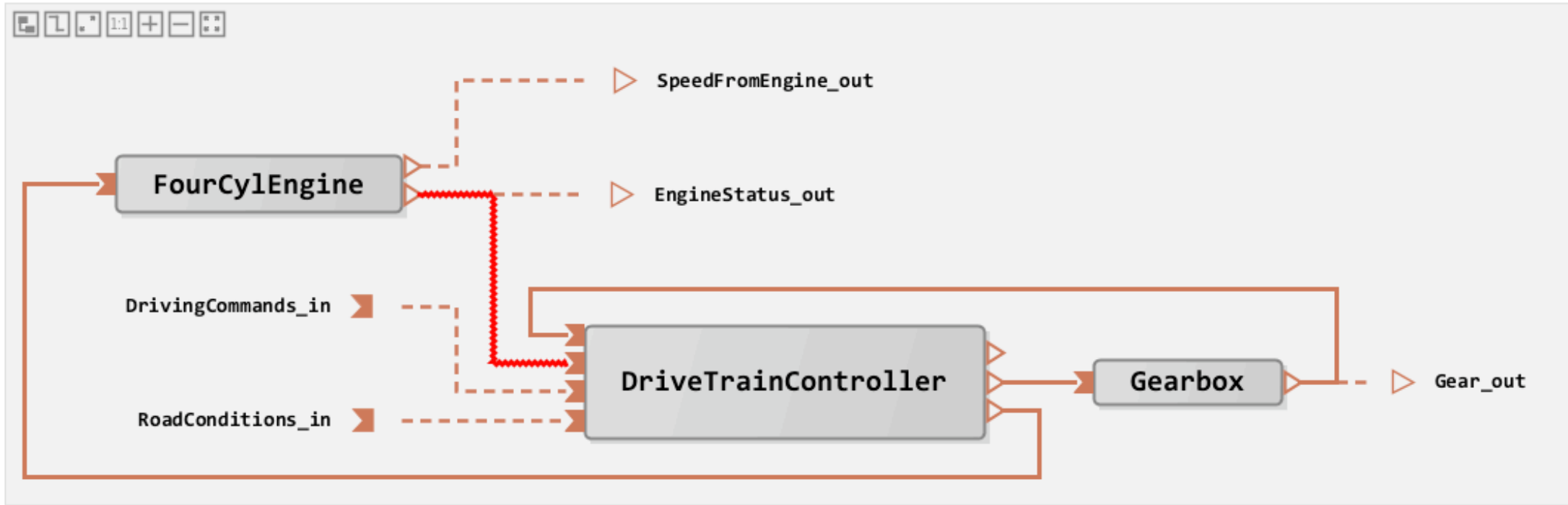
SPEX	ACCS	TIME	EQUIP	KNOW	eRAP
PRFC	RMAC	HRS	STRD	RSTR	Beyond high
PRFC	RMAC	HRS	STRD	RSTR	Beyond high
Independent: Very low					
PRFC	RMAC	HRS	STRD	RSTR	
PRFC	RMAC	HRS	STRD	RSTR	

SZ.30
Authenticity WebsiteData
RAP: Very low

SPEX	ACCS	TIME	EQUIP	KNOW	eRAP

Name	Title	Security goal classes	SUD Elements	Criteria				Depends on
				SAF	FIA	LAW	QUA	
SZ.16	Confidentiality UpdateRequest	Confidentiality	UpdateRequest			DPI		SZ.49 && SZ.37
SZ.17	Availability UpdateRequest	Availability	UpdateRequest				NSD	SZ.18 && SZ.50
SZ.18	Authenticity UpdateRequest	Authenticity	UpdateRequest			DPE		SZ.51
SZ.19	Confidentiality UpdateContainer	Confidentiality	UpdateContainer		LLO	DPE		SZ.21 && SZ.37 && SZ.52 && SZ.49
SZ.20	Availability UpdateContainer	Availability	UpdateContainer				NSD	SZ.21 && SZ.38 && SZ.50 && SZ.53
SZ.21	Authenticity UpdateContainer	Authenticity	UpdateContainer	ISL			NSD	SZ.39 && SZ.51 && SZ.54
SZ.22	Confidentiality ConfigurationProfiles	Confidentiality	ConfigurationProfiles			DPE		SZ.24 && SZ.52 && SZ.55
SZ.23	Availability ConfigurationProfiles	Availability	ConfigurationProfiles				NSD	SZ.24 && SZ.53 && SZ.56
SZ.24	Authenticity ConfigurationProfiles	Authenticity	ConfigurationProfiles				NSD	SZ.54 && SZ.57
SZ.26	Availability D4g	Availability	ProfileSelection				NSD	SZ.56
SZ.27	Authenticity ProfileSelection	Authenticity	ProfileSelection				NSD	SZ.57
SZ.28	Confidentiality WebsiteData	Confidentiality	WebsiteData			DPE		SZ.30 && SZ.40
SZ.29	Availability WebsiteData	Availability	WebsiteData				NSD	SZ.30 && SZ.41
SZ.30	Authenticity WebsiteData	Authenticity	WebsiteData			DPE	NSR	SZ.42
SZ.52	Confidentiality GCU	Confidentiality	GCU		LMO	DPE		SZ.54
SZ.53	Availability GCU	Availability	GCU				NFV	SZ.54
SZ.54	Integrity GCU	Integrity	GCU	ISL	LMO	DPE	NFV	<no expr>

```
public functional component DriveTrain {
  produces SpeedFromEngine
  produces EngineStatus
  produces Gear where it < gearsCount
  consumes RoadConditions
  param int gearsCount
  consumes DrivingCommands
```

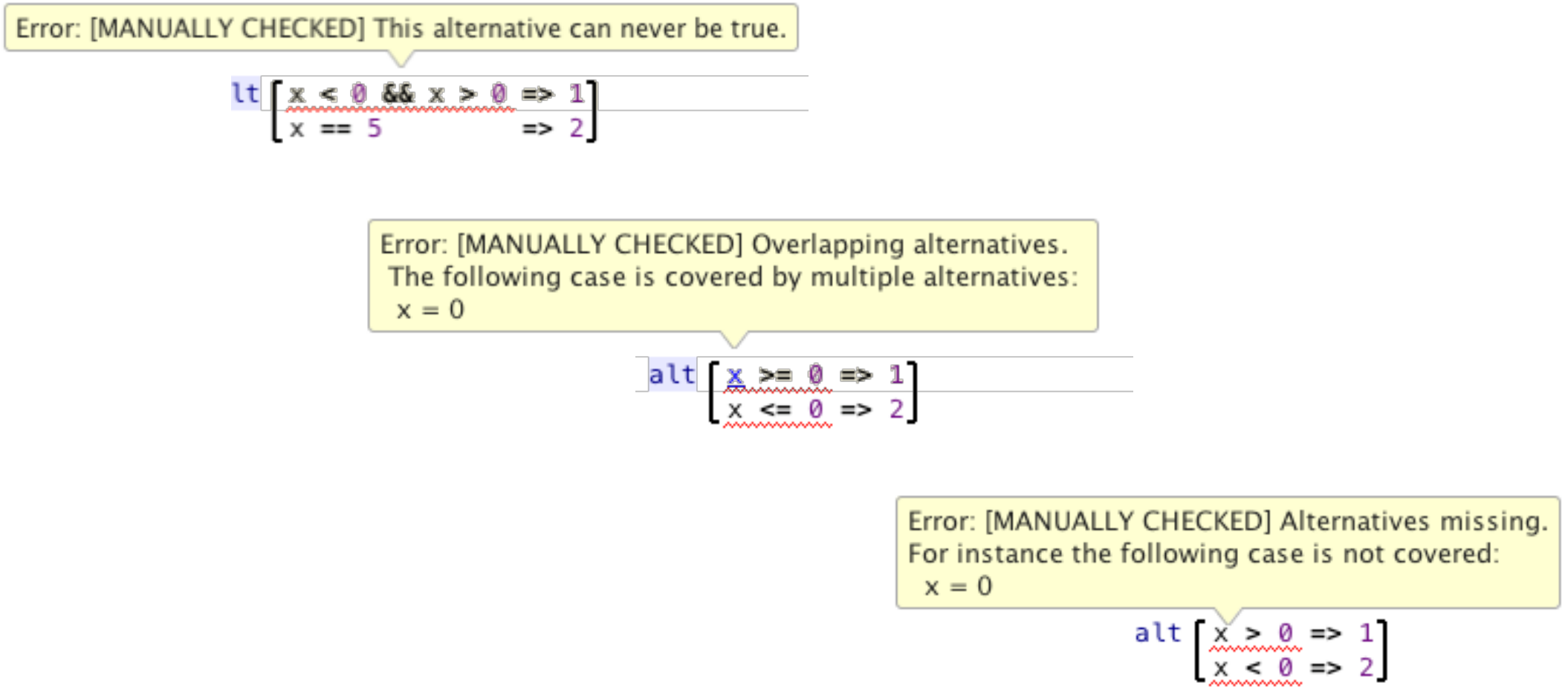



```
}
```

```
vector<int16, 3> aVector = [1; 2; 3] * 512;

vector<int16, 3> resultOfCrossProduct = aVector x aVector;

matrix<int16, 2x3> aMatrix = [1 + 2   2 * 7   42; 3   51   24];

matrix<int16, 3x2> transposedMatrix = aMatrix^T;
```

```
int32 averageIntArray(int32[] arr,int32 size) {
                 size
                 ∑   arr[i]
                i = 0
   return ------------------;
                 size
} averageIntArray (function)
```

„It‘s not just about code gen.“



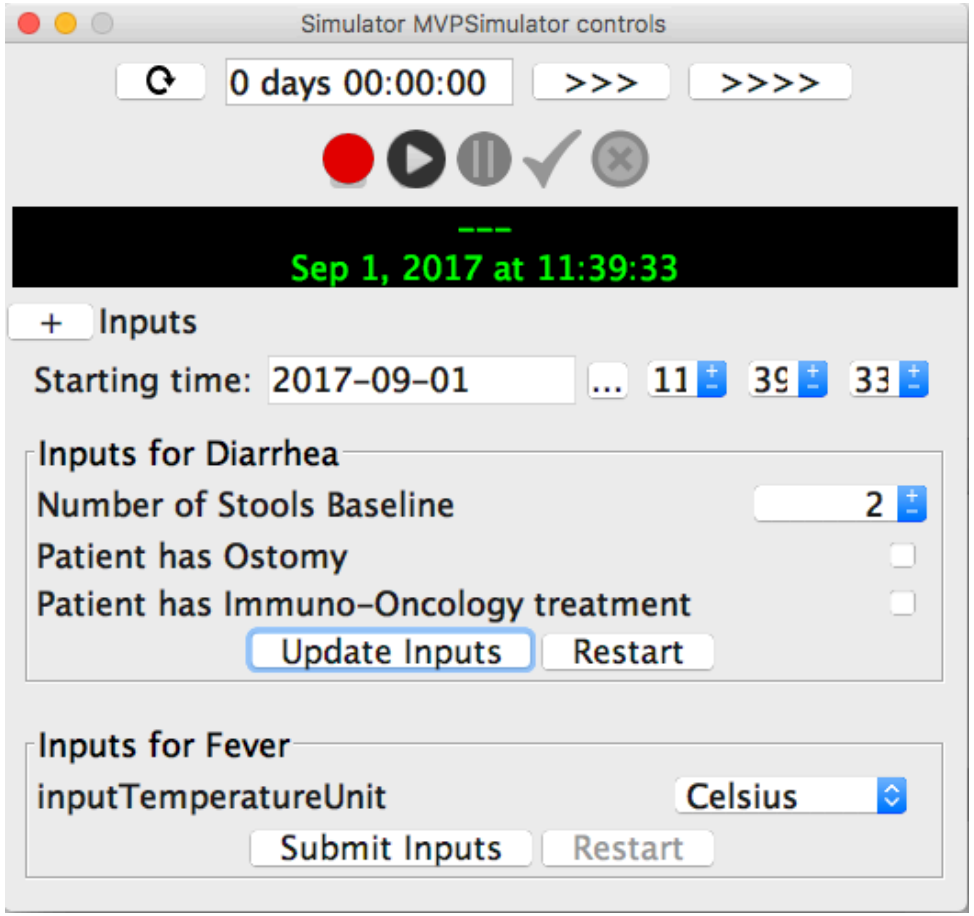
Analysis: SMT Solving of Expressions

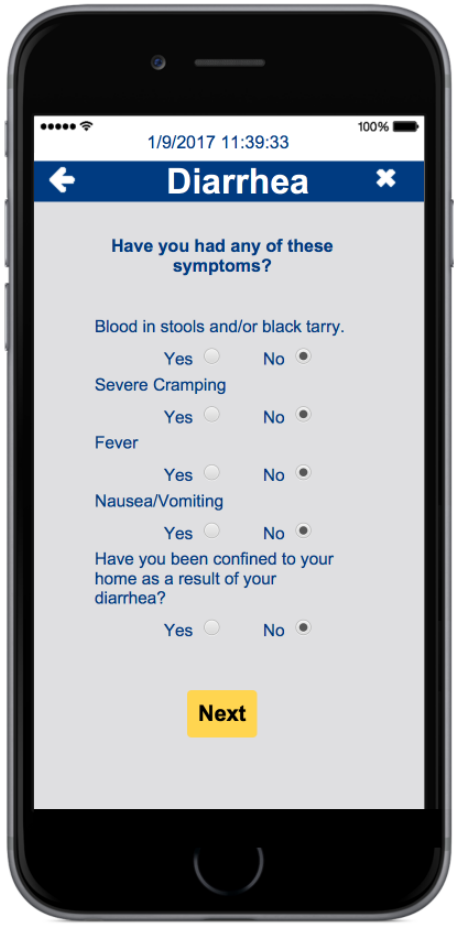
”It‘s not just about code gen.“

```
PASS
function test gradeStools
  given  7  expected  3
  given  6  expected  2
  given  5  expected  2
  given  4  expected  2

PASS
function test DiarrheaStoolsDecisionTree
  given  false, 1, true, false   expected   DiarrheaUSRecoLevel1Symptom
  given  false, 9, false, false  expected   DiarrheaUSRecoGrade3

PASS
function test checkScreeningQuestion
  given   answers to DiarrheaScreeningQuestionnaire{   expected  true
            dietarySupplements: false
            medication        : true
            hospitalized      : false
          }
```

Test: Run Tests directly in the IDE

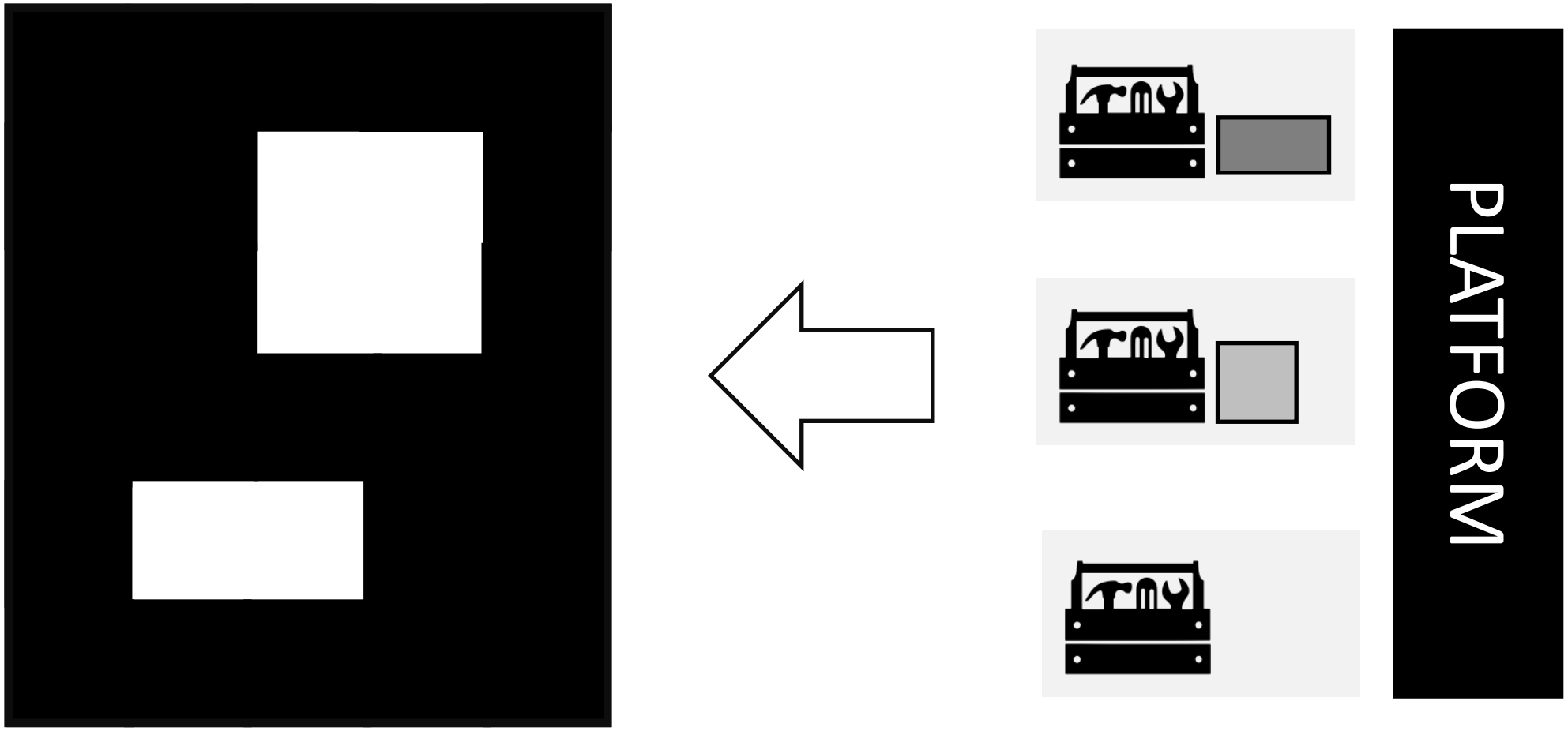
„It‘s not just about code gen.“





Validate: Simulate the program

CONSTRUCTION



DSLs are the Tools

Understand Domain

Design & Implement Language + IDE

Validate with Users

Generator/Interpreter

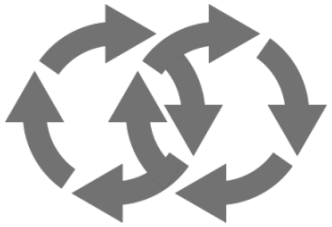
Deploy / Use

Why DSLs are the Tools

More Efficient Development

Meaningful Validation

Integrate Domain Experts



Decouple from Technology

DSLs are the Tools

Design & Implement Language + IDE

Generator/Interpreter

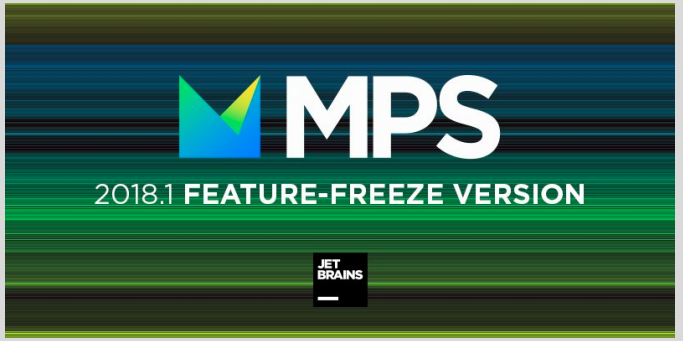
Language Workbench

DSLs are the Tools

Tools for efficiently building eco-systems of languages and their IDEs

Language Workbench

DSLs are the Tools



Extremely Powerful
Used in Practice
Open Source

Language Workbench

4

Expressing Variability with DSLs

1 Write each product from scratch

Languages are optimized for the domain
Syntax is concise, very little boilerplate
Just write every product from scratch.

[Works only for simple cases]
[Is at odds with SE best practices]

2 Use Variability Patterns in the DSL

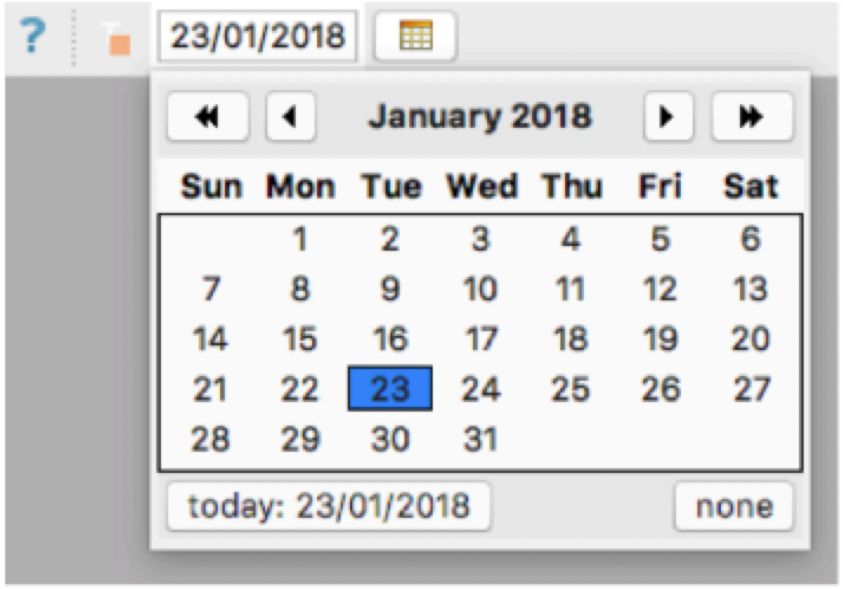
Design Variability into the Language
Similar to object-oriented programming

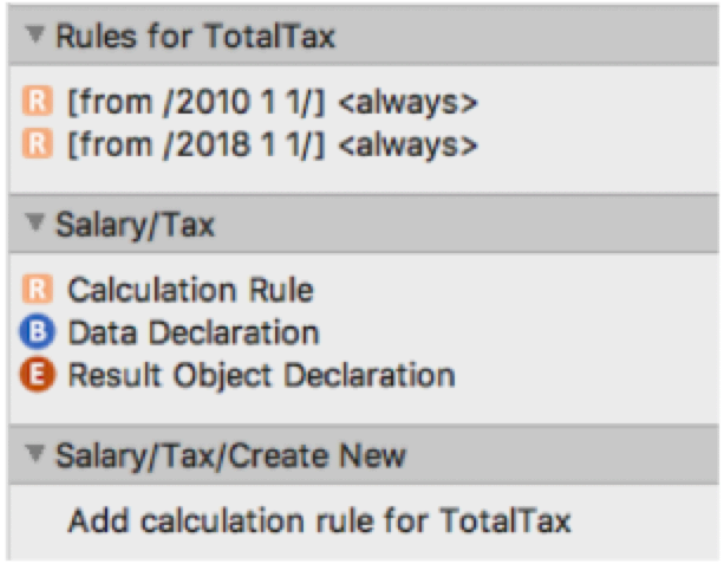
[Scales very well, esp. with verification]
[Might be harder to understand for DEs]

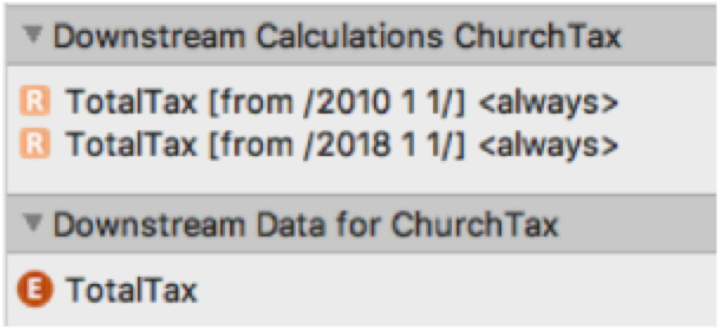
2 Use Variability Patterns in the DSL

```
calculation for Tax
  depends Salary as s
    valid from /2017 01 01/
  calculate [monthly] {
    ...
  }
```

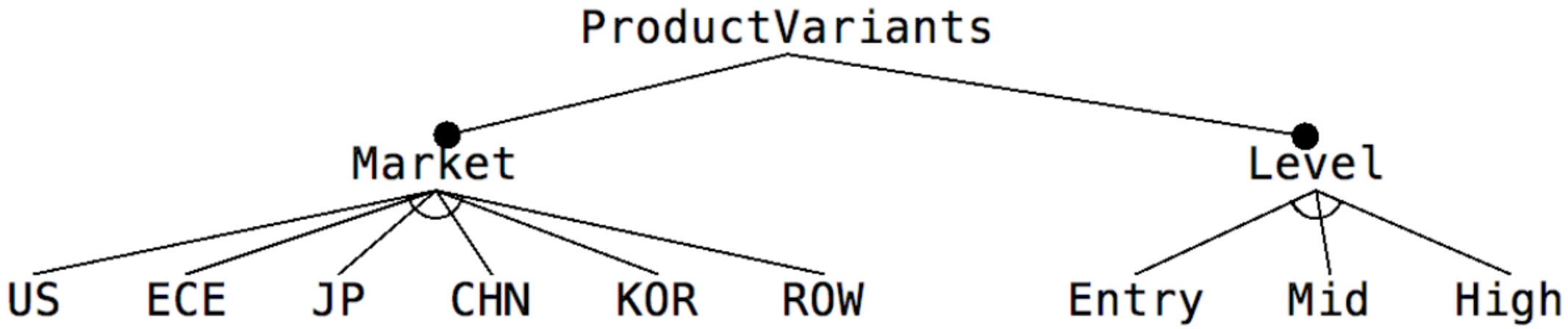
```
calculation for Tax
  depends Salary as s
          SomeOtherThing as t
    valid from /2017 07 01/
       if ctx.employment.person.homeAddress.state == BW
  calculate [monthly] { .. }
```





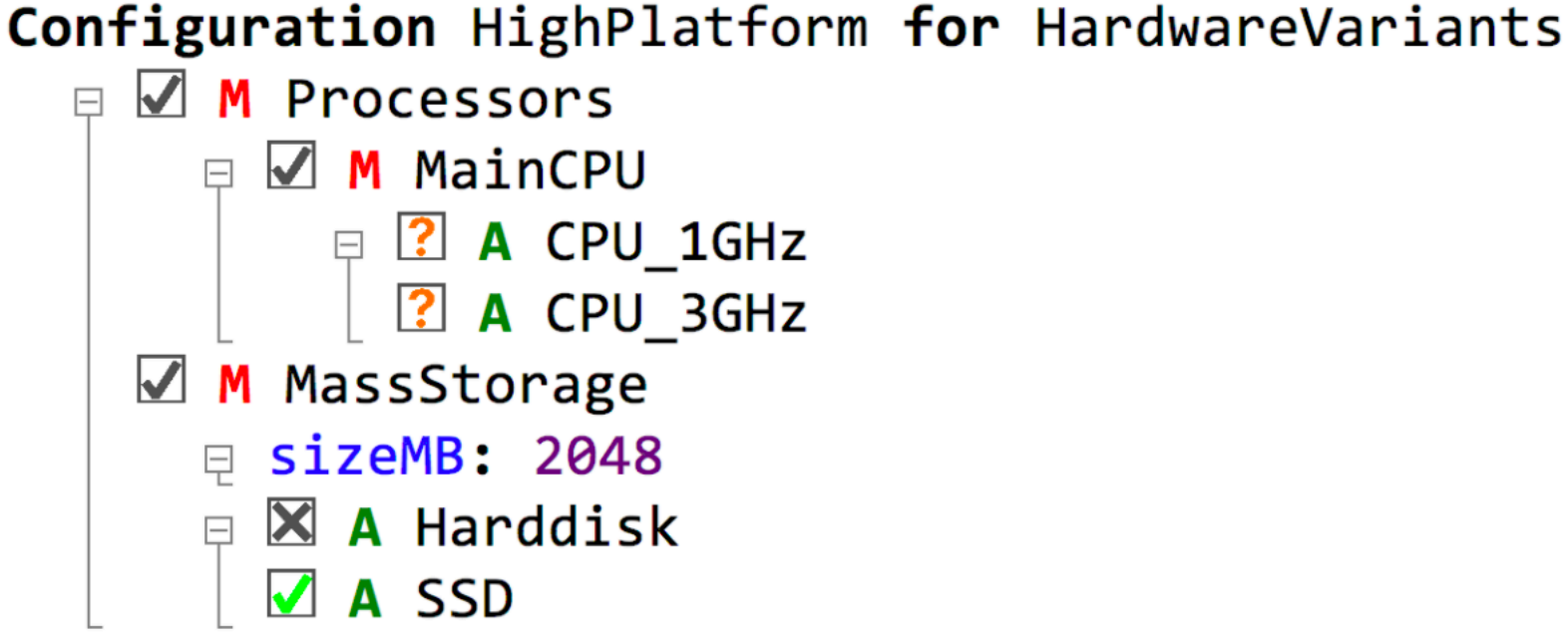


3 Overlay Configuration



Feature model for high-level variability

3 Overlay Configuration



Interactive, solver-backed configuration

3 Overlay Configuration

```
map_visible {
precondition Graphics::available
use 300
read MapDatabasePart :
```

Feature condition	Result
fm .pf .Map2Dim	15
fm .pf .Map2_5Dim	40
fm .pf .Map3Dim	80
default:	<no default>

```
MB via Harddisk.standard

}
```

Variation points in DSL code.

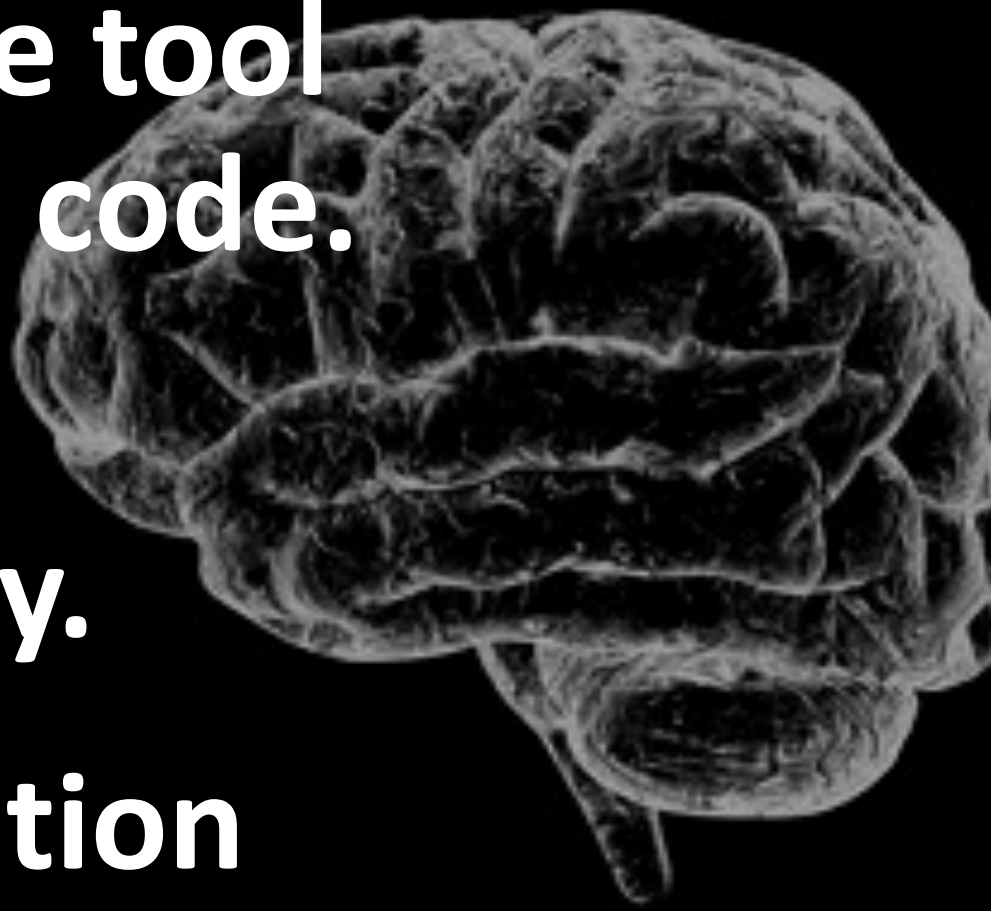
here: system modeling and performance simulation

5

Summary and Recommendations

Variability is more than Configuration

DSLs are a very productive tool
to create product-specific code.



Language Workbenches
let you create DSLs quickly.

You can overlay configuration
if needed and combine both approaches.